MODBUS Organization

MODBUS/TCP Server

지원 버전 TOP Design Studio V1.0 이상

CONTENTS

Touch Operation Panel을 사용해주시는 고객님께 감사 드립니다.

1. 시스템 구성

본 드라이버는 TOP가 MODBUS/TCP 서버 기능을 추가하여 동작하도록 합니다

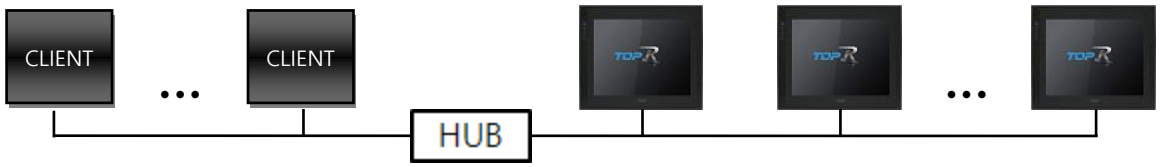
외부 장치	통신 방식	시스템 설정	케이블
MODBUS/TCP Client	Ethernet (TCP)	3. TOP 통신 설정	트위스트 페어 케이블 *주1)

*주1) 트위스트 페어 케이블

- STP(실드 트위스트 페어 케이블) 혹은 UTP(비실드 트위스트 페어 케이블) 카테고리 3, 4, 5 를 의미 합니다.
- 네트 워크 구성에 따라 허브, 트랜시버 등의 구성기기에 접속 가능하며 이 경우 다이렉트 케이블을 사용 하십시오.

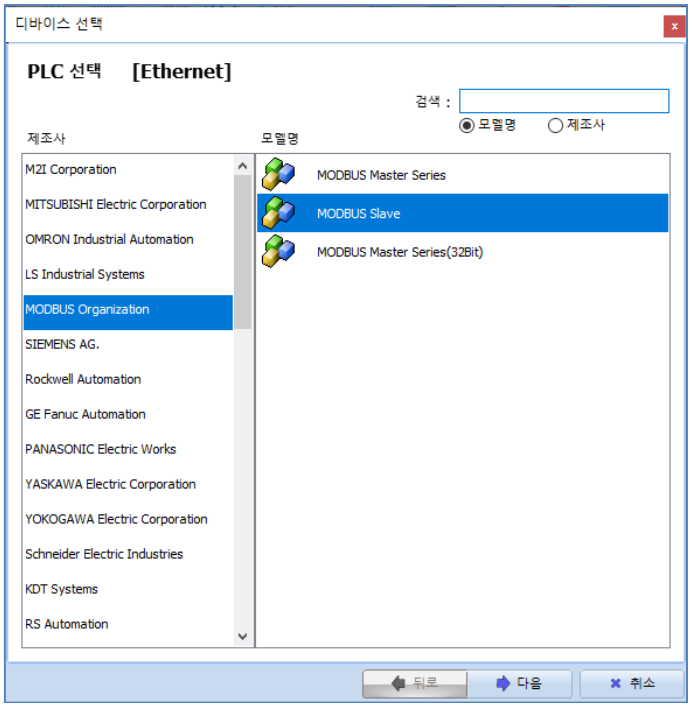
■ 연결 가능 구성

・N : N 연결



2. 외부 장치 선택

■ TOP 모델 및 포트 선택 후 외부 장치를 선택합니다.



설정 사항		내용
TOP	모델	TOP의 디스플레이와 프로세스를 확인하여 터치 모델을 선택합니다.
외부 장치	제조사	TOP와 연결할 외부 장치의 제조사를 선택합니다. "MODBUS Organization"를 선택 하십시오.
	PLC	TOP와 연결할 외부 장치를 선택 합니다. **모델** / **인터페이스** / **프로토콜** MODBUS Slave / Ethernet / MODBUS TCP 연결을 원하는 외부 장치가 시스템 구성 가능한 기종인지 1장의 시스템 구성에서 확인 하시기 바랍니다.

3. TOP 통신 설정

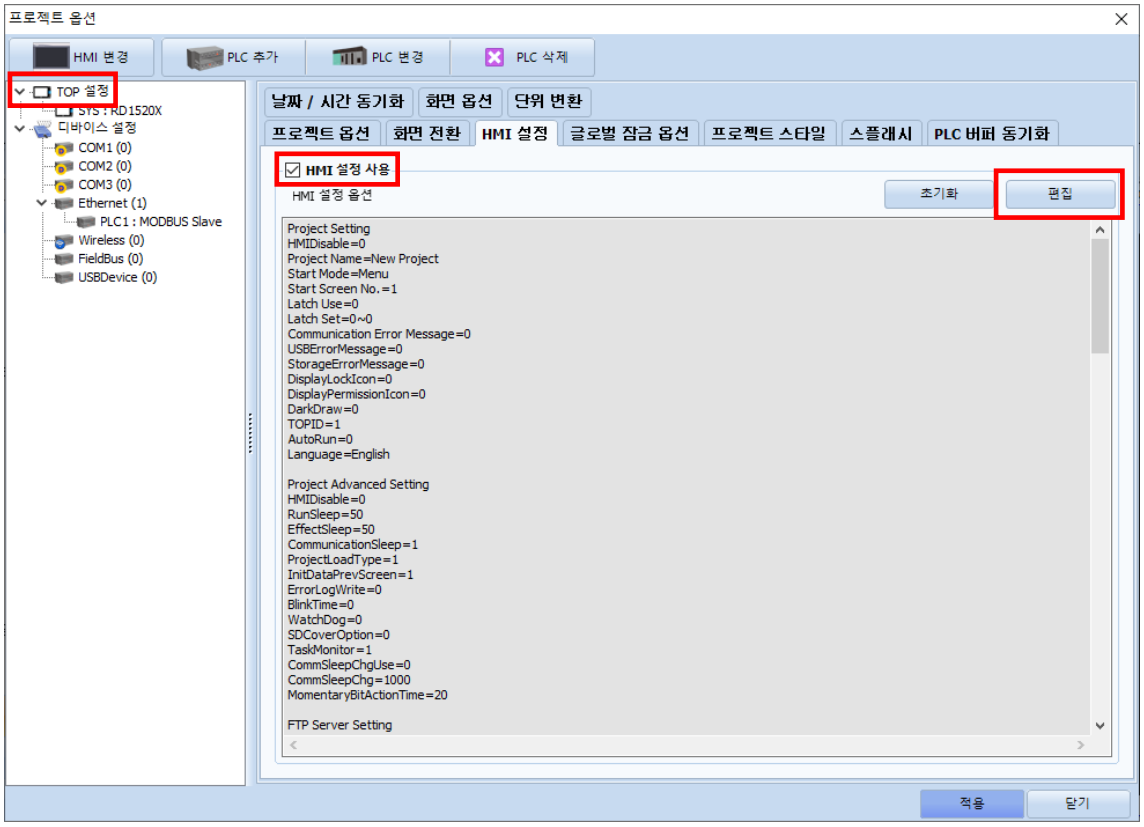
통신 설정은 TOP Design Studio 혹은 TOP 메인 메뉴에서 설정 가능 합니다. 통신 설정은 외부 장치와 동일하게 설정해야 합니다.

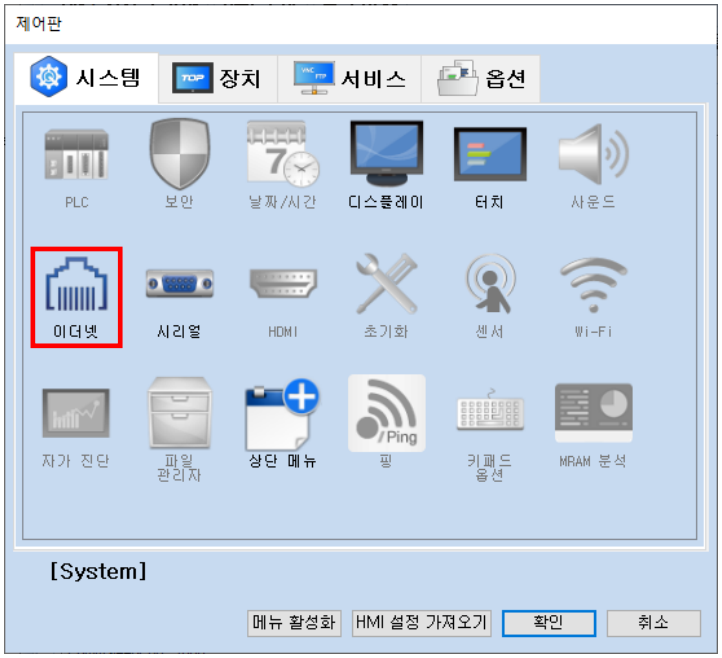
3.1 TOP Design Studio 에서 통신 설정

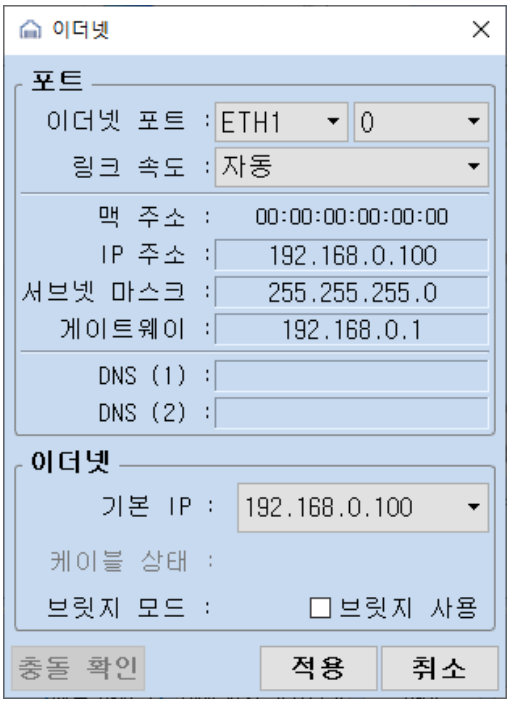
(1) 통신 인터페이스 설정

■ [프로젝트] → [속성] → [TOP 설정] → [HMI 설정] → [HMI 설정 사용 체크] → [편집] → [이더넷]

– TOP 통신 인터페이스를 TOP Design Studio에서 설정합니다.







항 목	TOP	외부 장치	비 고
IP 주소	192.168.0.100	192.168.0.50	
서브넷 마스크	255.255.255.0	255.255.255.0	
게이트 웨이	192.168.0.1	192.168.0.1	

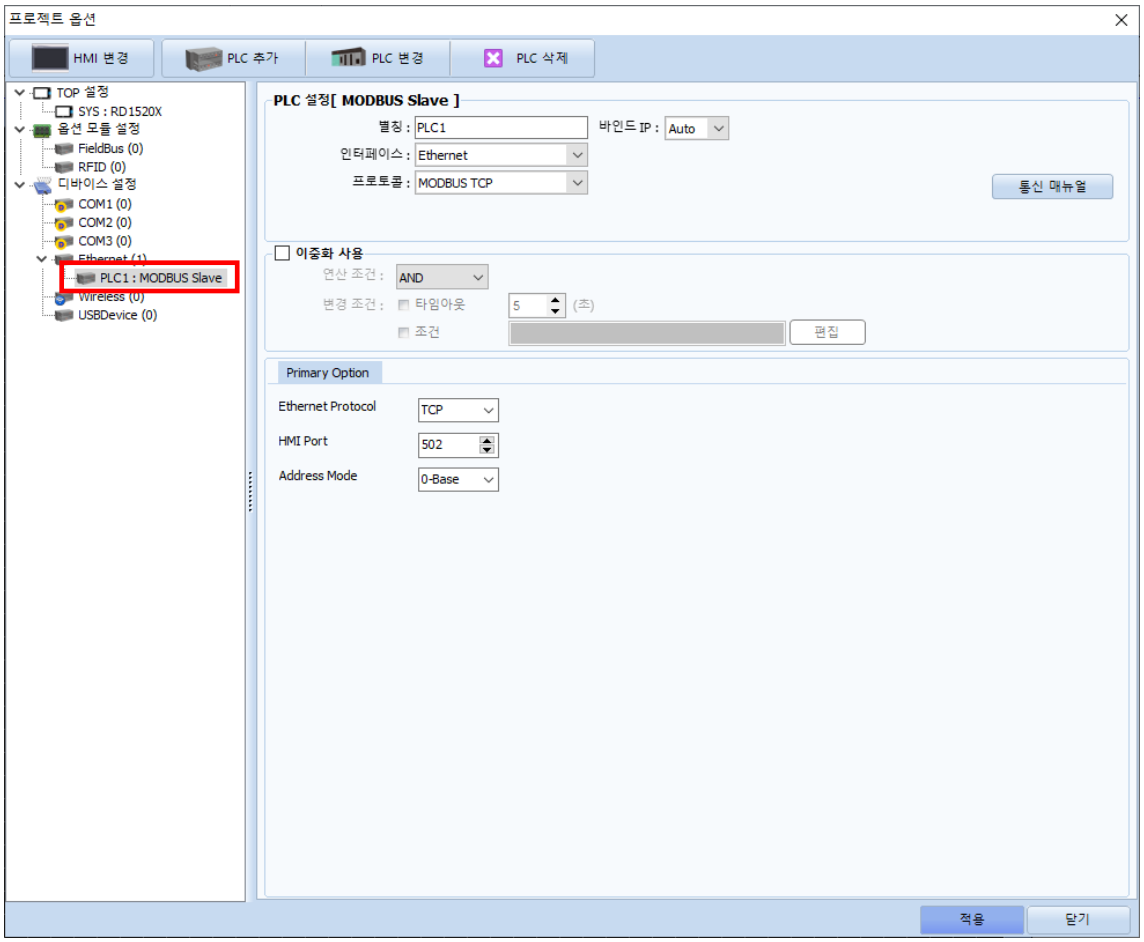
※ 위의 설정 내용은 본 사에서 권장하는 예제입니다.

항 목	설 명
IP 주소	TOP의 IP 주소를 설정합니다.
서브넷 마스크	네트워크의 서브넷 마스크를 입력합니다.
게이트 웨이	네트워크의 게이트 웨이를 입력합니다.

(2) 통신 옵션 설정

■ [프로젝트] → [프로젝트 속성] → [PLC 설정 > Ethernet > PLC1 : MODBUS Slave]

– MODBUS Slave 통신 드라이버의 옵션을 TOP Design Studio에서 설정합니다.



항 목	설 정	비 고
인터페이스	"Ethernet"을 선택합니다.	"2. 외부 장치 선택" 참고
프로토콜	TOP – 외부 장치 간 통신 프로토콜을 선택합니다.	
Ethernet Protocol	외부 장치의 IP 주소를 입력 합니다.	
HMI Port	TOP의 모드버스 통신 포트 번호를 설정합니다.	
Address Mode	모드버스 PDU Address의 -1 여부를 설정합니다.	*주1)

*주1) 클라이언트의 사양에 맞게 설정하십시오.

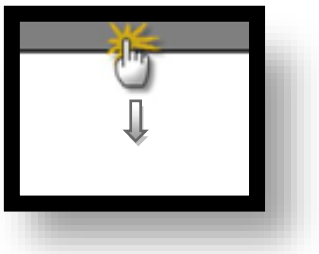
TOP의 SYS0 데이터를 읽기 위해 주소 0을 요청하면 0-Base 선택.

TOP의 SYS0 데이터를 읽기 위해 주소 1을 요청하면 1-Base 선택.

3.2 TOP에서 통신 설정

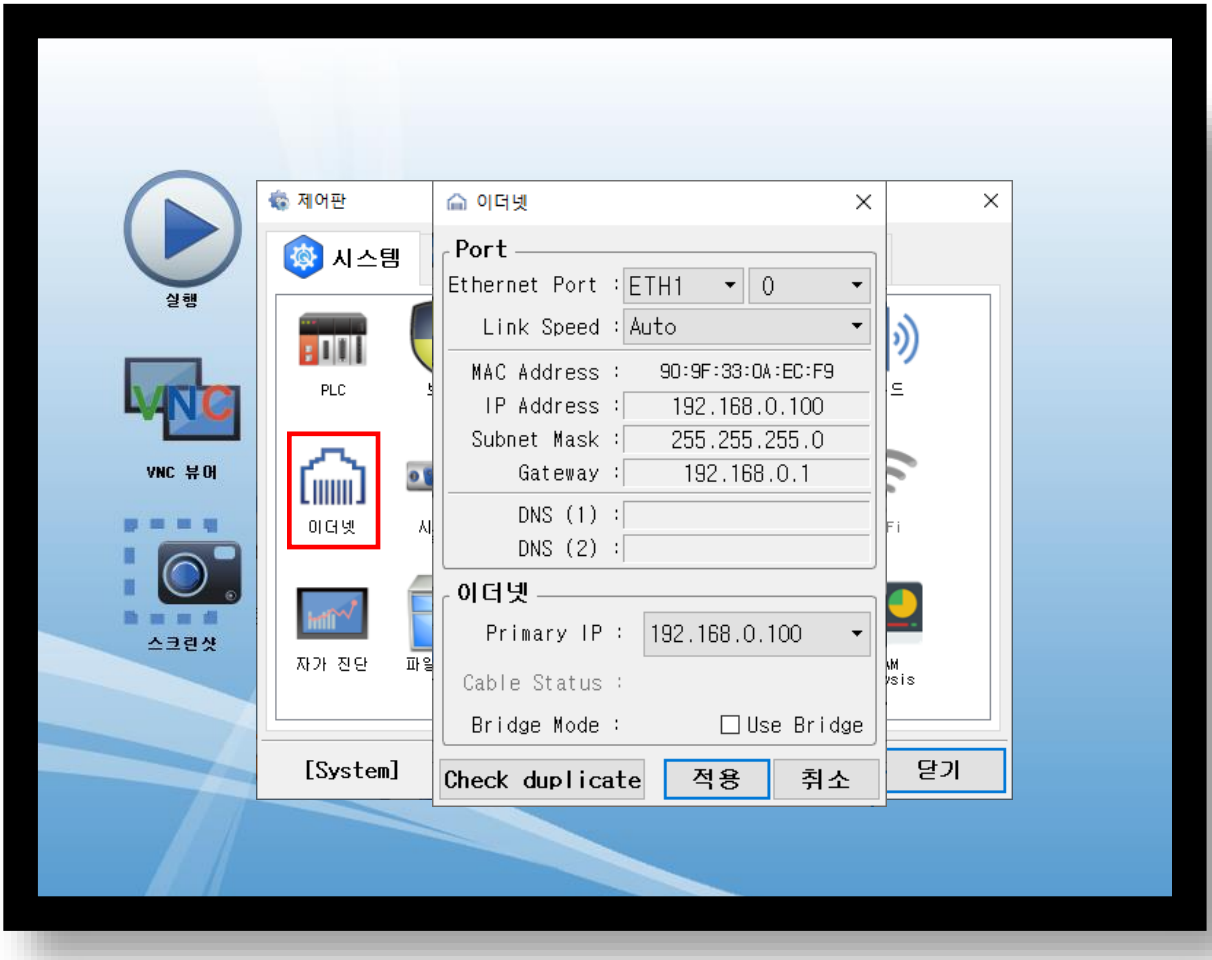
※ "3.1 TOP Design Studio 에서 통신 설정" 항목의 "HMI 설정 사용"을 체크 하지 않은 경우의 설정 방법입니다.

■ TOP 화면 상단을 터치하여 아래로 드래그 합니다. 팝업 창의 "EXIT"를 터치하여 메인 화면으로 이동합니다.



(1) 통신 인터페이스 설정

■ [제어판] → [이더넷]



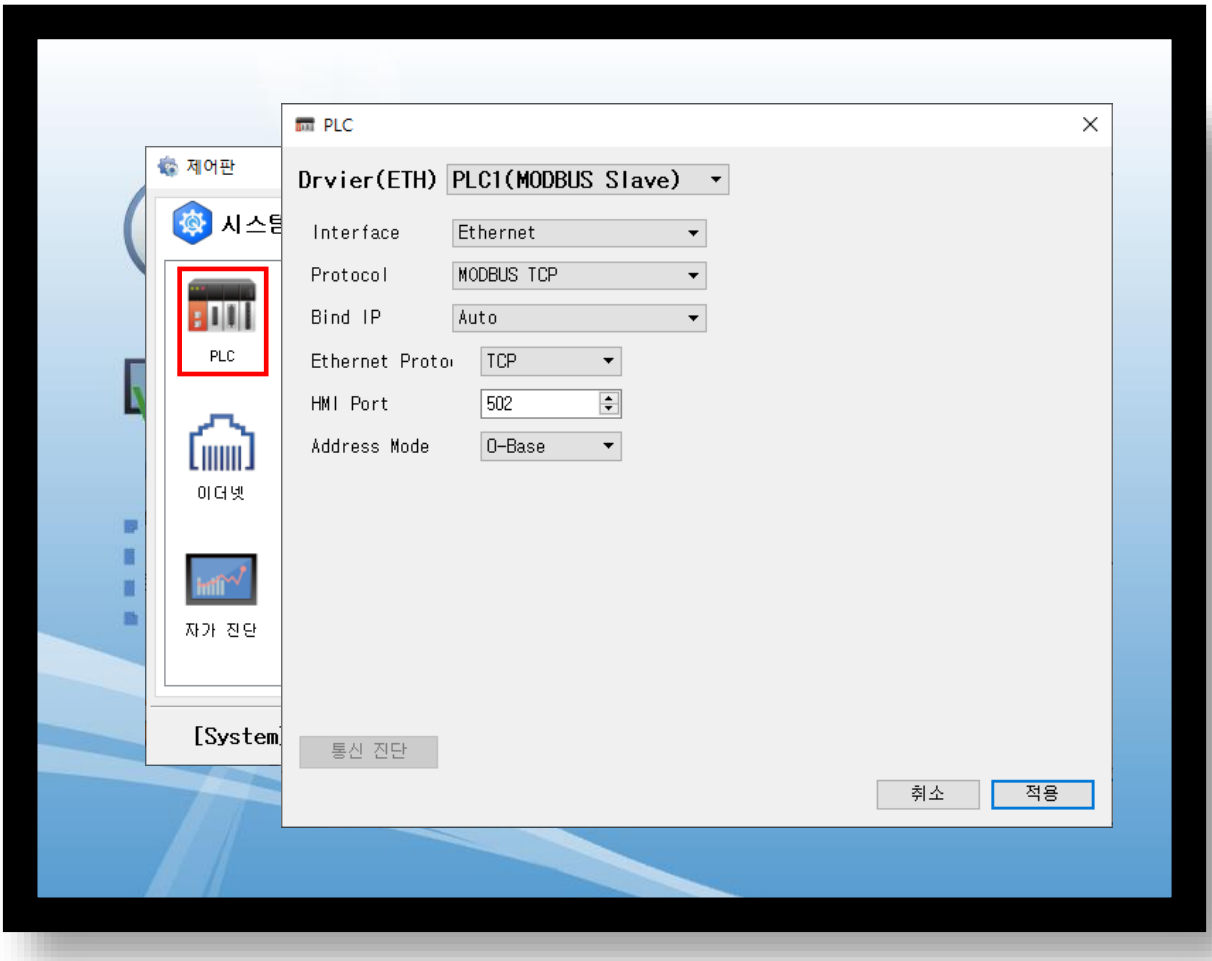
항 목	TOP	외부 장치	비 고
IP 주소	192.168.0.100	192.168.0.50	
서브넷 마스크	255.255.255.0	255.255.255.0	
게이트 웨이	192.168.0.1	192.168.0.1	

※ 위의 설정 내용은 본 사에서 권장하는 예제입니다.

항 목	설 명
IP 주소	TOP의 IP 주소를 설정합니다.
서브넷 마스크	네트워크의 서브넷 마스크를 입력합니다.
게이트 웨이	네트워크의 게이트 웨이를 입력합니다.

(2) 통신 옵션 설정

■ [제어판] → [PLC]



※ 위의 설정 내용은 본 사에서 권장하는 예제입니다.

항 목	설 정	비 고
인터페이스	"Ethernet"을 선택합니다.	"2. 외부 장치 선택" 참고
프로토콜	TOP – 외부 장치 간 통신 프로토콜을 선택합니다.	
Ethernet Protocol	외부 장치의 IP 주소를 입력 합니다.	
HMI Port	HMI의 모드버스 통신 포트 번호를 설정합니다.	
Address Mode	모드버스 PDU Address의 -1 여부를 설정합니다.	*주1)

*주1) 클라이언트의 사양에 맞게 설정하십시오.
TOP의 SYS0 데이터를 읽기 위해 주소 0을 요청하면 0-Base 선택.
TOP의 SYS0 데이터를 읽기 위해 주소 1을 요청하면 1-Base 선택.

3.3 통신 진단

본 드라이버는 통신 진단을 지원하지 않습니다.

클라이언트에서 접속과 데이터 읽기 요청을 시도하여 통신 연결을 확인하십시오.
주의) TOP가 실행(Run) 중이어야 합니다.

4. 지원 어드레스

TOP에서 지원하는 데이터에 대해 설명합니다.

주소	비트	워드	비고
SYS	0.0 ~ 10239.15	0 ~ 10239	*주1)

*주1) TOP-VIEW는 0~65535를 지원합니다.

※ TOP 내부 메모리 → 모드버스 데이터 모델링

TOP 내부 메모리를 모드버스 데이터로 표현하면 Holding Register에 해당합니다.
명령어 0x03으로 읽을 수 있으며 명령어 0x06, 0x10으로 값을 변경 할 수 있습니다.
Coil, Discrete Input, Input Register를 접근하는 명령어도 지원하지만
명령어가 다르더라도 결국 같은 메모리 영역(TOP 내부 메모리)을 접근합니다.

■ 지원 명령어

Code (hex)	Descriptions
01	Read Coils
02	Read Discrete Inputs
03	Read Holding Registers
04	Read Input Registers
05	Write Single Coil
06	Write Single Register
0F	Write Multiple Coils
10	Write Multiple Registers

Appendix A. MODBUS TCP

WHAT IS MODBUS?

The MODBUS protocol was developed in 1979 by Modicon, Incorporated, for industrial automation systems and Modicon programmable controllers. It has since become an industry standard method for the transfer of discrete/analog I/O information and register data between industrial control and monitoring devices. MODBUS is now a widely-accepted, open, public-domain protocol that requires a license, but does not require royalty payment to its owner.

MODBUS devices communicate using a master-slave (client-server) technique in which only one device (the Client(Master)) can initiate

transactions (called queries). The other devices (slaves/servers) respond by supplying the requested data to the master, or by taking the action requested in the query. A slave is any peripheral device (I/O transducer, valve, network drive, or other measuring device) which processes information and sends its output to the master using MODBUS. The Acromag I/O Modules form slave/server devices, while a typical master device is a host computer running appropriate application software. Other devices may function as both clients (masters) and servers (slaves).

Masters can address individual slaves, or can initiate a broadcast message to all slaves. Slaves return a response to all queries addressed to them individually, but do not respond to broadcast queries. Slaves do not initiate messages on their own, they only respond to queries from the master.

A master's query will consist of a slave address (or broadcast address), a function code defining the requested action, any required data, and an error checking field. A slave's response consists of fields confirming the action taken, any data to be returned, and an error checking field. Note that the query and response both include a device address, a function code, plus applicable data, and an error checking field. If no error occurs, the slave's response contains the data as requested. If an error occurs in the query received, or if the slave is unable to perform the action requested, the slave will return an exception message as its response (see MODBUS Exceptions). The error check field of the slave's message frame allows the master to confirm that the contents of the message are valid. Traditional MODBUS messages are transmitted serially and parity checking is also applied to each transmitted character in its data frame.

At this point, It's important to make the distinction that MODBUS itself is an application protocol, as it defines rules for organizing and interpreting data, but remains simply a messaging structure, independent of the underlying physical layer. As it happens to be easy to understand, freely available, and accessible to anyone, it is thus widely supported by many manufacturers.

WHAT IS MODBUS TCP/IP?

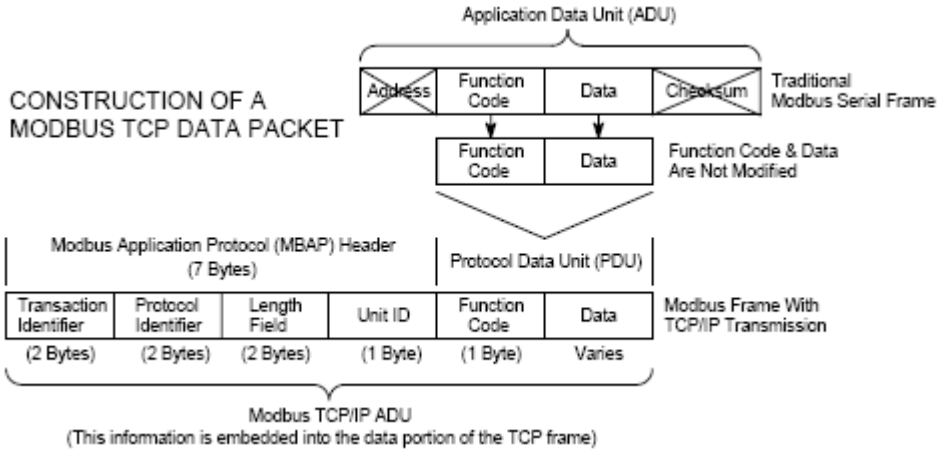
MODBUS TCP/IP (also MODBUS-TCP) is simply the MODBUS RTU protocol with a TCP interface that runs on Ethernet.
The MODBUS messaging structure is the application protocol that defines the rules for organizing and interpreting the data independent of the data transmission medium.

TCP/IP refers to the Transmission Control Protocol and Internet Protocol, which provides the transmission medium for MODBUS TCP/IP messaging.

Simply stated, TCP/IP allows blocks of binary data to be exchanged between computers. It is also a world-wide standard that serves as the foundation for the World Wide Web. The primary function of TCP is to ensure that all packets of data are received correctly, while IP makes sure that messages are correctly addressed and routed. Note that the TCP/IP combination is merely a transport protocol, and does not define what the data means or how the data is to be interpreted (this is the job of the application protocol, MODBUS in this case).

So in summary, MODBUS TCP/IP uses TCP/IP and Ethernet to carry the data of the MODBUS message structure between compatible devices. That is, MODBUS TCP/IP combines a physical network (Ethernet), with a networking standard (TCP/IP), and a standard method of representing data (MODBUS as the application protocol). Essentially, the MODBUS TCP/IP message is simply a MODBUS communication encapsulated in an Ethernet TCP/IP wrapper.

In practice, MODBUS TCP embeds a standard MODBUS data frame into a TCP frame, without the MODBUS checksum, as shown in the following diagram.



The MODBUS commands and user data are themselves encapsulated into the data container of a TCP/IP telegram without being modified in any way. However, the MODBUS error checking field (checksum) is not used, as the standard Ethernet TCP/IP link layer checksum methods are instead used to guaranty data integrity. Further, the MODBUS frame address field is supplanted by the unit identifier in MODBUS TCP/IP, and becomes part of the MODBUS Application Protocol (MBAP) header (more on this later).

From the figure, we see that the function code and data fields are absorbed in their original form. Thus, a Modbus TCP/IP Application Data Unit (ADU) takes the form of a 7 byte header (transaction identifier + protocol identifier + length field + unit identifier), and the protocol data unit (function code + data). The MBAP header is 7 bytes long and includes the following fields:

- **Transaction/invocation Identifier (2 Bytes):** This identification field is used for transaction pairing when multiple messages are sent along the same TCP connection by a client without waiting for a prior response.
- **Protocol Identifier (2 bytes):** This field is always 0 for MODBUS services and other values are reserved for future extensions.
- **Length (2 bytes):** This field is a byte count of the remaining fields and includes the unit identifier byte, function code byte, and the data fields.
- **Unit Identifier (1 byte):** This field is used to identify a remote server located on a non TCP/IP network (for serial bridging). In a typical MODBUS TCP/IP server application, the unit ID is set to 00 or FF, ignored by the server, and simply echoed back in the response.

The complete MODBUS TCP/IP Application Data Unit is embedded into the data field of a standard TCP frame and sent via TCP to well-known system port 502, which is specifically reserved for MODBUS applications. MODBUS TCP/IP clients and servers listen and receive MODBUS data via port 502.
We can see that the operation of MODBUS over Ethernet is nearly transparent to the MODBUS register/command structure. Thus, if you are already familiar with the operation of traditional MODBUS, then you are already very with the operation of MODBUS TCP/IP.

A.1 "0" Device (Coil)

(1) Read Single Coil : 01

MASTER 기기에서 Slave 기기 측(국번:17번)의 "**0**00020–**0**00056 Coil" 데이터를 읽어 오는 예제를 통해 "01"명령어 프레임을 설명 합니다.

■ RTU Mode

(Master → Slave : 요청 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 규격)	명령어	전두디바이스		디바이스점수	
	H	L	H	L	H	L			H	L	H	L
Hex	00	01	00	00	00	06	11	01	00	13	00	25

(Slave → Master : 응답 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 규격)	명령어	데이터 개수(byte)	데이터			
	H	L	H	L	H	L				Coils 27~20	Coils 35~28	Coils 43~36	Coils 51~44	Coils 56~52
										L	–	–	–	H
Hex	00	01	00	00	00	08	11	01	05	CD	6B	B2	0E	1B

■ Coils 데이터 상태

Coils	27	26	25	24	23	22	21	20
on/off	1	1	0	0	1	1	0	1
Coils	35	34	33	32	31	30	29	28
on/off	0	1	1	0	1	0	1	1
Coils	43	42	41	40	39	38	37	36
on/off	1	0	1	1	0	0	1	0
Coils	51	50	49	48	47	46	45	44
on/off	0	0	0	0	1	1	1	0
Coils	59	58	57	56	55	54	53	52
on/off	–	–	–	1	1	0	1	1

0: OFF /

(2) Force Single Coil : 05

MASTER 기기에서 Slave 기기 측의 Coil **0**00173 에 FORCE "ON" 하는 예제를 통해 "05"명령어 프레임을 설명 합니다.

■ RTU Mode

(Master → Slave : 요청 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 규격)	명령어	전두디바이스		Force data	
	H	L	H	L	H	L			H	L	H	L
Hex	00	02	00	00	00	06	11	05	00	AC	FF	00

■ Force Data

	High	Low
Force ON	FF_H	00_H
Force OFF	00_H	00_H

(Slave → Master : 응답 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 규격)	명령어	전두디바이스		Force data	
	H	L	H	L	H	L			H	L	H	L
Hex	00	02	00	00	00	06	11	05	00	AC	FF	00

A.2 "1" Device (Discrete Input)

(1) Read Input Status : 02

MASTER 기기에서 Slave 기기 측(국번:17번)의 "**1**00197–**1**00218 Input" 데이터를 읽어 오는 예제를 통해 "02"명령어 프레임을 설명합니다.

(Master → Slave : 요청 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 국번)	명령어	선두디바이스		디바이스점수	
	H	L	H	L	H	L			H	L	H	L
Hex	00	03	00	00	00	06	11	02	00	C4	00	16

(Slave → Master : 응답 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 국번)	명령어	데이터 개수(byte)	데이터(Inputs)		
	H	L	H	L	H	L				10204~10197	10212~10205	10218~10213
Hex	00	03	00	00	00	06	11	02	03	AC	DB	35

■ Coils 데이터 상태

Coils	204	203	202	201	200	199	198	197
on/off	1	0	1	0	1	1	0	0
Coils	212	211	210	209	208	207	206	205
on/off	1	1	0	1	1	0	1	1
Coils	220	219	218	217	216	215	214	213
on/off	–	–	1	1	0	1	0	1

0: OFF / 1:ON

A.3 "3" Device (Input Register)

(1) Read Input Registers : 04

MASTER 기기에서 Slave 기기 측(국번:17번)의 "**3**00009 Register" 데이터를 읽어 오는 예제를 통해 "03"명령어 프레임을 설명 합니다.

(Master → Slave : 요청 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 국번)	명령어	선두디바이스		디바이스점수 (Word Count)	
	H	L	H	L	H	L			H	L	H	L
Hex	00	04	00	00	00	06	11	04	00	08	00	01

(Slave → Master : 응답 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 국번)	명령어	데이터 개수(byte)	데이터 30009 Register	
	H	L	H	L	H	L				H	L
Hex	00	04	00	00	00	05	11	04	02	00	0A

A.4 "4" Device (Holding Register)

(1) Read Holding Registers : 03

MASTER 기기에서 Slave 기기 측(국번:17)의 "**4**00108 – **4**00110 Register" 데이터를 읽어 오는 예제를 통해 "03"명령어 프레임을 설명 합니다.

(Master → Slave : 요청 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 국번)	명령어	선두디바이스		디바이스점수	
	H	L	H	L	H	L			H	L	H	L
Hex	00	05	00	00	00	06	11	03	00	6B	00	03

(Slave → Master : 응답 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 국번)	명령어	데이터 개수(byte)	데이터					
										40108 Register		40109 Register		40110 Register	
	H	L	H	L	H	L				H	L	H	L	H	L
Hex	00	05	00	00	00	09	11	03	06	02	2B	00	00	00	64

(2) Preset Single Register : 06

Slave 기기 측의 **4**00002 Register 에 00 03 (hex) 데이터를 입력 하는 예제를 통해 "06"명령어 프레임을 설명 합니다.

(Master → Slave : 요청 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 국번)	명령어	선두디바이스		Preset data	
	H	L	H	L	H	L			H	L	H	L
Hex	00	06	00	00	00	06	11	06	00	01	00	03

(Slave → Master : 응답 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 국번)	명령어	선두디바이스		Preset data	
	H	L	H	L	H	L			H	L	H	L
Hex	00	06	00	00	00	06	11	06	00	01	00	03

(3) Preset Multiple Register : 10

Slave 기기 측의 **4**00002 Register 에 "00 0A (hex)", "01 02 (hex)" 연속한 두 개의 데이터를 입력 하는 예제를 통해 "10"명령어 프레임을 설명 합니다. (Error Code : 90_H)

(Master → Slave : 요청 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 국번)	명령어	선두디바이스		Quantity of Register (Word Count)		데이터 갯수(Byte)	데이터			
														Register 40002		Register 40003	
	H	L	H	L	H	L			H	L	H	L		H	L	H	L
Hex	00	07	00	00	00	0B	11	10	00	01	00	02	04	00	0A	01	02

(Slave → Master : 응답 프레임)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave 국번)	명령어	선두디바이스		Quantity of Register (Word Count)	
	H	L	H	L	H	L			H	L	H	L
Hex	00	07	00	00	00	06	11	10	00	01	00	02